Mini Project Report

Topic 9: Linear Regression – Deep Learning - AI

Group 20

I. Abstract

This report illustrates a mini project that implements a linear regression model for stock price predictions, inspired by the ELR-MR (Evaluation Linear Regression based Machine Learning) model proposed by J. Margaret Sangeetha and K. Joy Alfia [1]. The project aims to replicate and extend the methodology outlined in the original study, providing a practical demonstration based on real stock data of Meta Platforms, Inc in the last 5 years. By using historical stock price data, the linear regression model attempts to predict future prices based on various pre-determined variables such as open, close, low, high and volume. This project not only validates the findings of the original research but also explores the model's robustness and accuracy through additional datasets and performance metrics. As the original paper has stated the dependent relationship that the volume factor has with other parameters, this project set out to further explore the correlation between them through using linear regression model as well. This project not only validates the findings of the original research but also explores the model's robustness and accuracy through additional datasets and performance metrics with comparation with the model provided in the scikit-learn library in Python. The results indicate that while linear regression offers a straightforward and interpretable approach to stock price prediction, its effectiveness is highly dependent on the assumption of linearity about the stock price trend.

II. Introduction

The research paper by J. Margaret Sangeetha and K. Joy Alfia has proposed the use of Evaluated Linear Regression model for stock market forecast. In this project the detail algorithm of the paper is designed and examined. Further evaluation metrics are used to confirm the accuracy of the implemented model.

III. Group Member and Assignment

- Dinh Ngoc Lap Thanh (Leader) 20226000:
  - Coordinate and allocate tasks.
  - Manage progress and review source code.
  - Write project report.
- Nguyen Duc Anh 20226009:

    - Design project presentation.

1. Stock price prediction model.
    - Trinh Thi Thuy Duong 20226034:
        - Develop stock price predicting model and cross examine results with Python's built-in class `LinearRegression()` from sklearn library.
    - Doan Thi Thu Quyen 20226063:
        - Compute model evaluation metrics and design visual summaries of model results.
2. Stock trading volume prediction model.
    - Duong Phuong Thao 20226001:
        - Develop stock volume predicting model and cross examine results with Python's built-in class `LinearRegresison()` from sklearn library.
        - Compute model evaluation metrics and visual summaries of model results.
    - Vu Ngoc Dung 20226032:
        - Review Thao's code.

IV. Evaluated Linear Regression.

1. Stock price prediction model.

1.1. Key information.

- Data source: Yahoo finance through utilizing `yfinance` library.

```
data =  GetFacebookInformation.history(period="5y")
vectors = data[['Open', 'Close', 'High', 'Low']].values.tolist()
```

- Number of observations use: 1258 entries spanning 5 years, with 80% used for training and 20% for testing.
- Independent parameters: open, close, low, high stock price.
- Dependent parameter: the following day stock price

1.2. Algorithm detail

- We design a linear regression model that assumes that:

$$y_{i+1} = \beta + \alpha_1 open_1 + \alpha_2 close_i + \alpha_3 low_i + \alpha_4 high_i + E$$

$$\widehat{y_{i+1}} = \frac{(open_{i+1} + close_{i+1} + low_{i+1} + high_{i+1})}{4}$$

- Since our model includes a constant term β, we will a column of 1s to our data.
- Our aim is to choose a list of parameters' weight α that minimizes the sum of squared errors using stochastic gradient descent, where the model parameter is updated using the average gradient of a batch of 1 data points at each iteration.

$$SSE = \sum_{i=1}^{n} (y_i - Y_i)^2 \qquad (1)$$

$$y_i: predicted\ result\ for\ each\ data\ point$$

$$Y_i: empirical\ result\ for\ each\ data\ point$$

- Our error function is defined as the difference between the predicted value and the true value.
- Explanation for the model main function `least_square_fit`:
  - `xs`: List of vectors, each vector represents the input features for a single data point.
  - `ys`: List of floats, representing target value for each data point.
  - `learning_rate`: The size of the steps taken to reach the minimum for each iteration of gradient descent.
  - `num_steps`: number of iterations.
  - `batch_size`: number of entries considered for each "batch" to compute the gradient (stochastic gradient descent).
  - We set our initial guess for α to be 0. With each iteration of `num_steps`, the inner loop through the list of vectors `xs` in batches, each of size "`batch_size`", creating `batch_xs` and `batch_ys`.
  - `sqerror_gradient(x,y,guess)` computes the gradient of the squared error for a single data point `(x,y)` with respect to the current guess by calculating the derivative for the respective data point in equation (1) while `vector_mean` calculates the mean gradient over the batch, in this case the two consecutive data points.

$$gradient = \frac{dSSE_{(x_i,y_i)}}{dx} = 2ex_i = 2(y_i - Y_i)x_i$$

  - `gradient_step` updates the current guess using the computed gradient and the learning rate. The updated guess moves in the direction of the negative gradient to minimize the error. Finally, the function returns the final guess for the regression coefficient after all iterations are complete.

1.3. Results and analyzation

1.3.1. Claim

- The following results were conducted with data from 29-5-2019 to 28-5-2024, each entry recorded at 04:00 AM Eastern Time.

1.3.2. Model results and explanations.

- With a `learning_rate` of 1^{-8} and number of iterations 10000, we found the model coefficient and intercept, the result is show in Figure 1.

```
[17]: learning_rate = 0.00000001
beta = least_squares_fit(train_vectors, train_y_values, learning_rate, 10000, 1)
print("Nghiệm tìm được bằng least_squares_fit:", beta)

Nghiệm tìm được bằng least_squares_fit: [-0.1631814723040778, 0.6482664870968531, 0.22609599418399337, 0.2887174356893341, 0.02675270575340634]
```

Figure 1: The predicting model calculated coefficients.

- Compared with the given model `LinearRegression()` provided by the sklearn library, the conducted model share a similarity in parameters' weights, as seen in Figure 2.

```
[18]: model = LinearRegression()
new_vectors = train_vectors
for vector in new_vectors:
    vector.pop()
model.fit(new_vectors, train_y_values)
coef = model.coef_.tolist()
coef.append(model.intercept_)
print("Hệ số của các biến:", coef)

Hệ số của các biến: [-0.26489412806554463, 0.6949658283124249, 0.18476839965680897, 0.38363938145177734, 1.0179334493052181]
```

Figure 2: The sklearn library's model calculated coefficients.

- Figure 3 shows that as we increase the number of iterations, our model's coefficients and intercept converging to those of the benchmark.

```
✓ 19m 8.0s
Nghiệm tìm được bằng least_squares_fit: [-0.1631814723040778, 0.6482664870968531, 0.22609599418399337, 0.2887174356893341, 0.02675270575340634]
Nghiệm tìm được bằng least_squares_fit: [-0.23352948639752477, 0.7046530976180898, 0.22333675960197336, 0.3055602824280842, 0.05136795586865181]
Nghiệm tìm được bằng least_squares_fit: [-0.24907850398905304, 0.7088386808127399, 0.22544349038284892, 0.314801159547922, 0.0756545480566027]
Nghiệm tìm được bằng least_squares_fit: [-0.25508638713437787, 0.7058892860586174, 0.22817837208921415, 0.3209615887474396, 0.09968128813904378]
Nghiệm tìm được bằng least_squares_fit: [-0.2588210824359192, 0.7026255488861253, 0.23055657042846636, 0.3255042554976628, 0.12344612447733767]

from sklearn.linear_model import LinearRegression
model = LinearRegression()
new_vectors = train_vectors
for vector in new_vectors:
    vector.pop()
model.fit(new_vectors, train_y_values)
coef = model.coef_.tolist()
coef.append(model.intercept_)
print("Hệ số của các biến:", coef)

Hệ số của các biến: [-0.2637895586855289, 0.6942013127858819, 0.18579046742487632, 0.3822672267115382, 1.0073020257203495]
```

Figure 3: The predicting model coefficients as num_steps change from 10000 to 50000 with a leap of 10000 between each iteration vs sklearn model coefficient.

- Still, the `num_steps` of 10000 proof to be usable with the following evaluation metrics and as the benefit in increasing `num_steps` don't outweigh the cost in runtime, we will set it as 10000 and continue.
- As for the `learning_rate`, our testing shows as any `learning_rate` that's larger than 1^{-8} results in NaN coefficients.

1.3.3. Evaluation metrics and model comparison

a. Correlation Matrix

- Table 1 shows the coefficient of correlation between every two features.

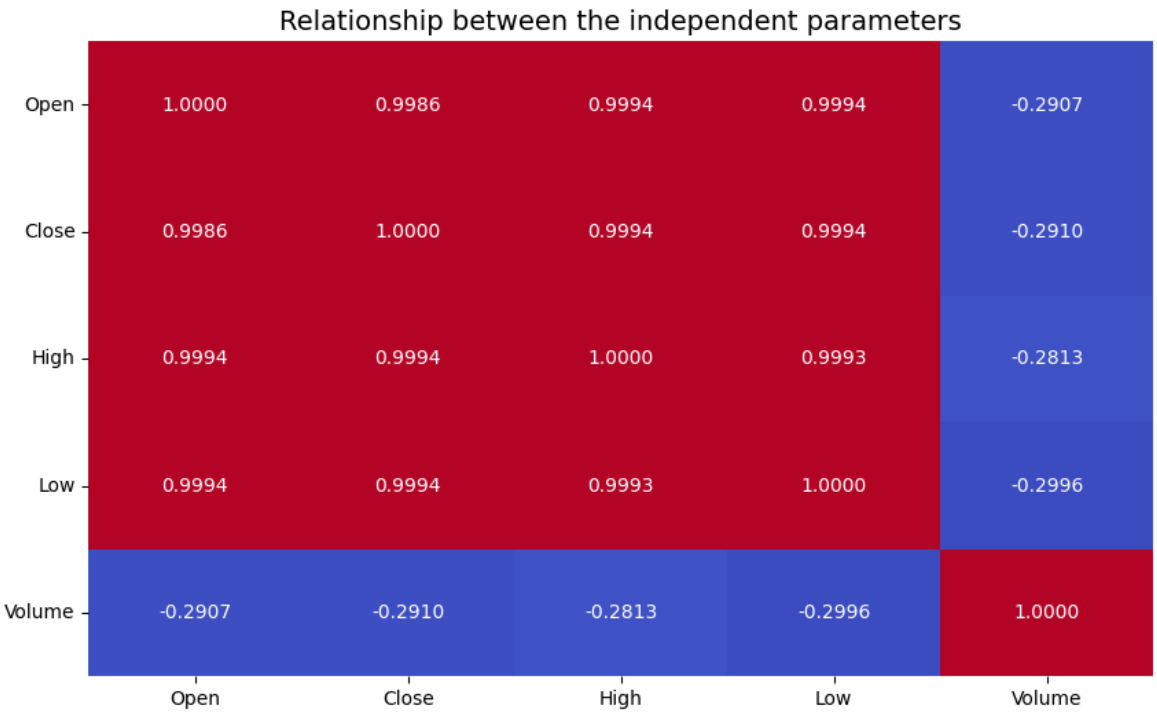

Table 1: Relationship between the independent parameters.

- The results confirm the original findings that the feature volume is dependent on open, close, low, and high, and thus, we exclude it from our final calculations
- Furthermore, as open, close, low, and high are highly dependent on each other, it is imperative that we take all of them into consideration when designing the algorithm.
- With the number of iterations limited to 10000, the final evaluation metrics nearly approach those of sklearn's.

	Model	R-Squared	Adjusted R-Squared	Multiple R	Standard Error	MSE
0	Custom Least Squares	0.991322	0.991181	0.995651	7.730271	59.757094
1	Sklearn LinearRegression	0.991405	0.991266	0.995693	7.693209	59.185463

Table 2: Achieved regression statistics by ELR-ML technique analysis.

- Figure 4 shows a considerable similarity between the estimation of two models.

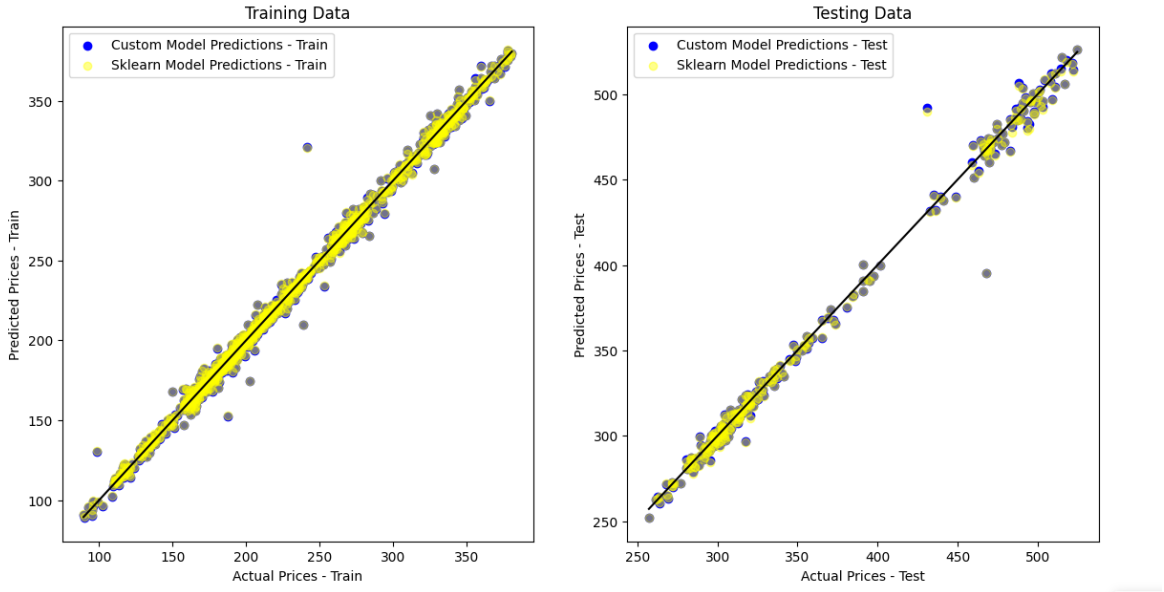

Figure 4: Plot of the predictive performance of stock market forecast between two models.

2. Stock volume prediction model.

2.1. Key information

- Data source: Yahoo finance through utilizing `yfinance` library.

```
# Tải dữ liệu lịch sử của cổ phiếu (ví dụ: 'META' cho Meta)
ticker = 'META'
data = yf.download(ticker, period='5y')
# Tính giá trung bình
data['Price'] = data[['Open', 'Close', 'High', 'Low']].mean(axis=1)

# Chọn các cột cần thiết
data = data[['Open', 'High', 'Low', 'Close', 'Volume', 'Price']]
data
```

- Number of observations use: 1258 entries spanning 5 years.
- Independent parameters: price (mean of open, close, low, high).
- Dependent parameters: the following day stock volume.

- We store our data in a csv file and using pandas, convert it into a DataFrame and select required columns in the form of numpy array, thereby allowing convolution operation between arrays, which will come handy later.

2.2. Algorithm detail

- We design a linear regression model that assumes that:

$$volume_{i+1} = b + \frac{a(open_i + close_i + high_i + low_i)}{4} + E$$

- Our aim is to choose lists of parameters' weight a and b that minimize the sum of squared errors using gradient descent, where the model parameter is updated using the average gradient of a batch of 1 data points at each iteration.

$$SSR = \frac{\sum_{i=1}^{n}(y_i - Y_i)^2}{n}$$

$$y_i: Predicted\ result\ for\ data\ point\ i$$
$$Y_i: Empirical\ result\ for\ data\ point\ i$$

- Explanation for the model main function `predict`:
  - `alpha`: The size of the steps taken to reach the minimum for each iteration of gradient descent.
  - `Epochs`: number of iterations.
  - `m`: number of entries.
  - `a`: slope of each data point.
  - `b`: intercept of each data point
  - `volumes_pred`: predicted volume arrays.
  - `error`: difference between the predicted volume and empirical volume.
  - `db`: the derivative of data point's following volume(predict) w.r.t b.
  - `da`: the derivative of data point's following volume(predict) w.r.t a.

  → We set our initial guess for a and b to be random numbers. With each iteration, the inner loop will update the predicted volume for all data points then find the gradients `da` and `db` by taking the derivate of MSE equation and move a, b along the negative gradient. After finishing the inner loop, the function will predict the following volume with the same formula to calculate `volumes_pred`.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - Y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(ax_i + b - Y_i)^2$$

$$a' = \frac{1}{n}\sum_{i=1}^{n} 2(ax_i + b - Y_i)^1 \, x_i$$

$$b' = \frac{1}{n}\sum_{i=1}^{n} 2(ax_i + b - Y\_i)^1$$

2.3. Results and analyzation

2.3.1. Claim

- The following results were conducted with data from 29-5-2019 to 28-5-2024, each entry recorded at 04:00 AM Eastern Time.

2.3.2. Model results and explanations.

- With a learning rate of 0.02 and number of epochs 100000, we found the model slope, intercept and the SSR, the result is shown in Figure 5&6.

```
5]: # Dự đoán volume sử dụng các hệ số vừa tìm được
    train_volumes_pred, train_incepter, train_slope = predict(train_prices, train_volumes)

Hệ số hồi quy (intercept): 24320989.46322063
Hệ số hồi quy (slope): -4752602.253755248
```

```
# Tính tổng bình phương của các phần dư hậu hồi quy (SSR)
ssr = mean_squared_error(train_volumes, train_volumes_pred) * len(train_volumes)
print(f"Tổng bình phương của các phần dư (SSR): {ssr}")

Tổng bình phương của các phần dư (SSR): 2.547382667584343e+17
```

Figure 5&6. The predicting model's calculated coefficients

- Compared with the given model `LinearRegression()` provided by the sklearn library, the conducted model differ greatly in parameters' weights but the same SSR value.

```
Hệ số hồi quy (intercept) với sklearn: 40415075.804823965
Hệ số hồi quy (slope) với sklearn: -69404.40021444074
Tổng bình phương của các phần dư (SSR) với sklearn: 2.5473988882788134e+17
```

Figure 7. The sklearn library model's calculated coefficients

```
[ ]: ssr_sklearn - ssr

[ ]: 64.0
```

Figure 8. The difference of SSR value between sklearn library's model and DIY model.

- The `epoch` of 100000 is proof to be usable with the following evaluation metrics and as the benefit in increasing `epoch` don't outweigh the cost in runtime, we will set it as 100000 and continue.
- As for the `alpha`, our testing shows that any `alpha` that's larger proof to be inadequate in its evaluation metrics.

2.3.3. Evaluation metrics and comparison

- With the number of iterations limited to 100000, the final evaluation metrics equal to those of sklearn's.
- Figure 9 shows the results of both models on the test data.

```
Hệ số hồi quy (intercept): 19597409.56175294
Hệ số hồi quy (slope): -2418716.578976548
                     Model  R-Squared  Multiple R  Standard Error  \
0      Custom Least Squares   0.081895    0.286174    1.591285e+07
1  Sklearn LinearRegression   0.081895    0.286174    1.591285e+07

            MSE
0  2.532190e+14
1  2.532190e+14
```

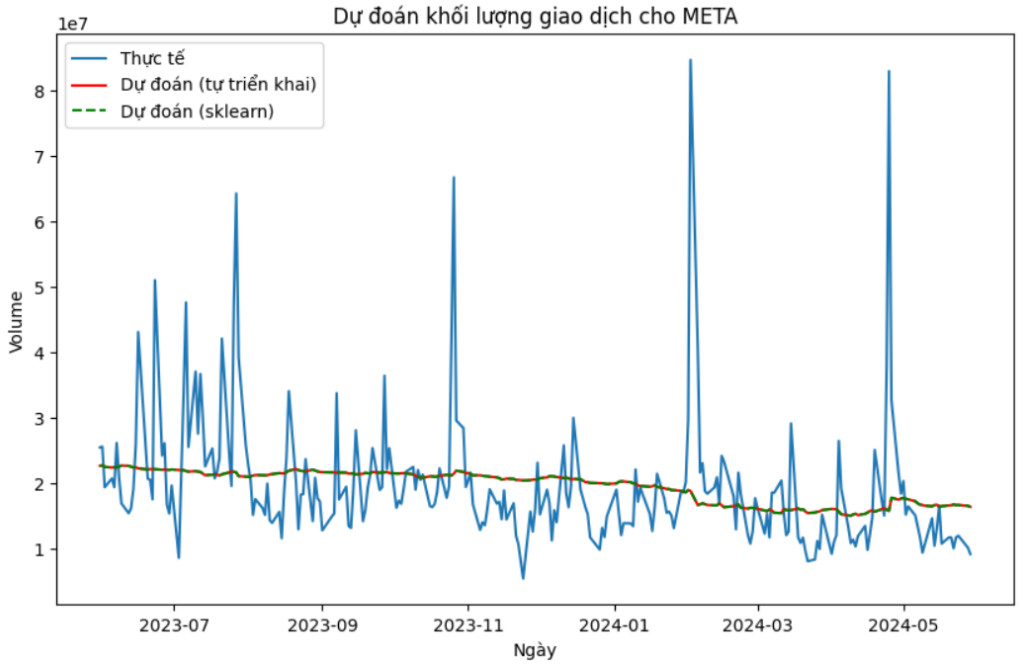

Figure 9: Plot the predictive performance of two models against the actual stock trends.

V. References

[1] https://doi.org/10.1016/j.measen.2023.100950